# *Self-Adaptive Discovery Mechanisms for Optimal Performance in Fault-Tolerant Networks*

**Kevin Mills and Doug Montgomery**

**DARPA FTN PI Meeting**
**July 29, 2001**

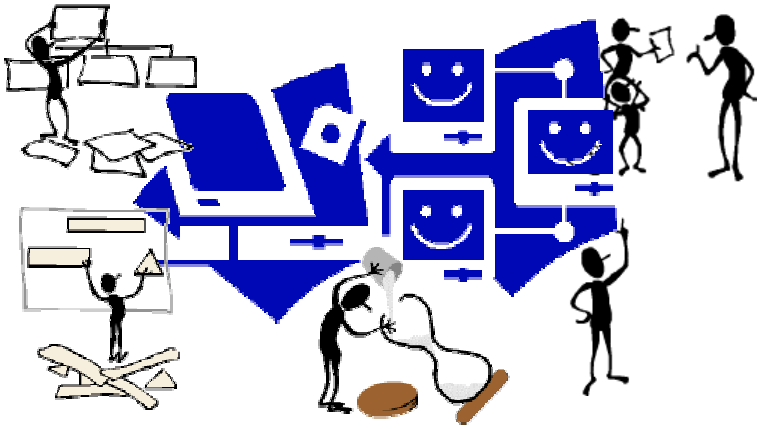
***Survivable Software for Harsh Environments***

# Presentation Outline

- Quick Introduction to the Project – Quad Chart
- Introduction to Dynamic Discovery Protocols
  - Discovery Protocols in Essence
  - Foundation for Fault-Tolerant Distributed Systems
- Related NIST Work on Discovery Protocols
  - Modeling, Analysis, and Measurement
  - Three Example Projects
- Details Regarding Our Fault-Tolerant Networks Project
  - Objective and Motivation
  - Example Problem from One Discovery Protocol
  - Selected Control Parameters and Adaptive Behaviors
  - Research Plan
- Conclusions

# Self-Adaptive Discovery Mechanisms for Optimal Performance in Fault-Tolerant Networks

***Survivable Software for Harsh Environments***



**Kevin Mills and Doug Montgomery NIST**

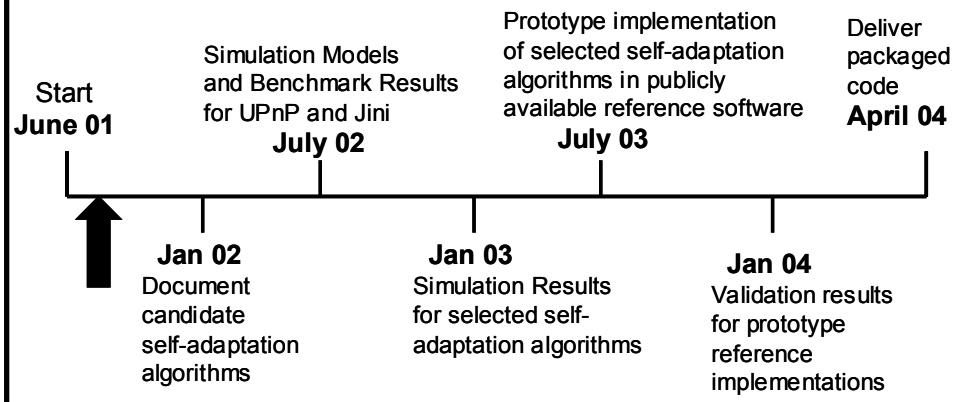
## Innovations

- Read-and-react decentralized discovery mechanisms that observe the network state, estimate and adjust relevant parameters, measure effects, and adapt as necessary
  - **Emerging industry discovery protocols assume parameters configured and tuned by hand and network topology evolves slowly. Assumptions invalid for most military scenarios (fast response, high mobility, cyber and physical attacks, and jamming of communication channels).**
- Simulation modeling techniques to create and measure the effects of continuous, controlled changes in network topologies

## Impact

- Influence the design of next-generation discovery protocols so they can self-configure tunable parameters and can adjust in response to network dynamics
- Influence the design of next-generation simulation systems to provide better support for simulating controlled and continuous change in network topologies and for collecting related measurements

## Schedule

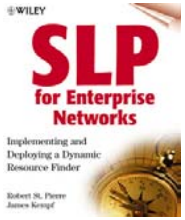


# Dynamic Discovery Protocols in Essence

Dynamic discovery protocols enable **network elements** (including software clients and services, as well as devices):

(1) to **discover** each other without prior arrangement,
(2) to **express** opportunities for collaboration,
(3) to **compose** themselves into larger collections that cooperate to meet an application need, and
(4) to **detect and adapt to changes** in network topology.

# Selected First-Generation Dynamic Discovery Protocols

| | | |
|---|---|---|
| JINI 3-Party Design | Universal Plug and Play 2-Party Design |  Adaptive 2/3-Party Design |
| The Salutation Consortium Vertically Integrated Design | HAVi Network-Dependent Design | Bluetooth™ Network-Dependent Design |

# Dynamic Discovery Protocols Provide Foundation for Fault-Tolerant Distributed Systems

- In the future, **all software systems will be distributed** systems written to operate over a network, where conditions vary.
- Dynamic **discovery protocols provide a foundation** upon which such distributed systems will be constructed.
- **Understanding** the current (first) generation of discovery protocols **essential** to enable the military to establish requirements and to help industry to improve designs for the second and subsequent generations.

# NIST Investigating Emerging Commercial Designs

- **Modeling and analyzing** the structure, behavior, performance and logical properties of **selected discovery protocols** (Jini, UPnP, and SLP)
- **Measuring** the **performance** properties of the current reference software that implements **selected discovery protocols** (Jini, UPnP, and SLP)

# Technical Approach to Modeling & Analysis

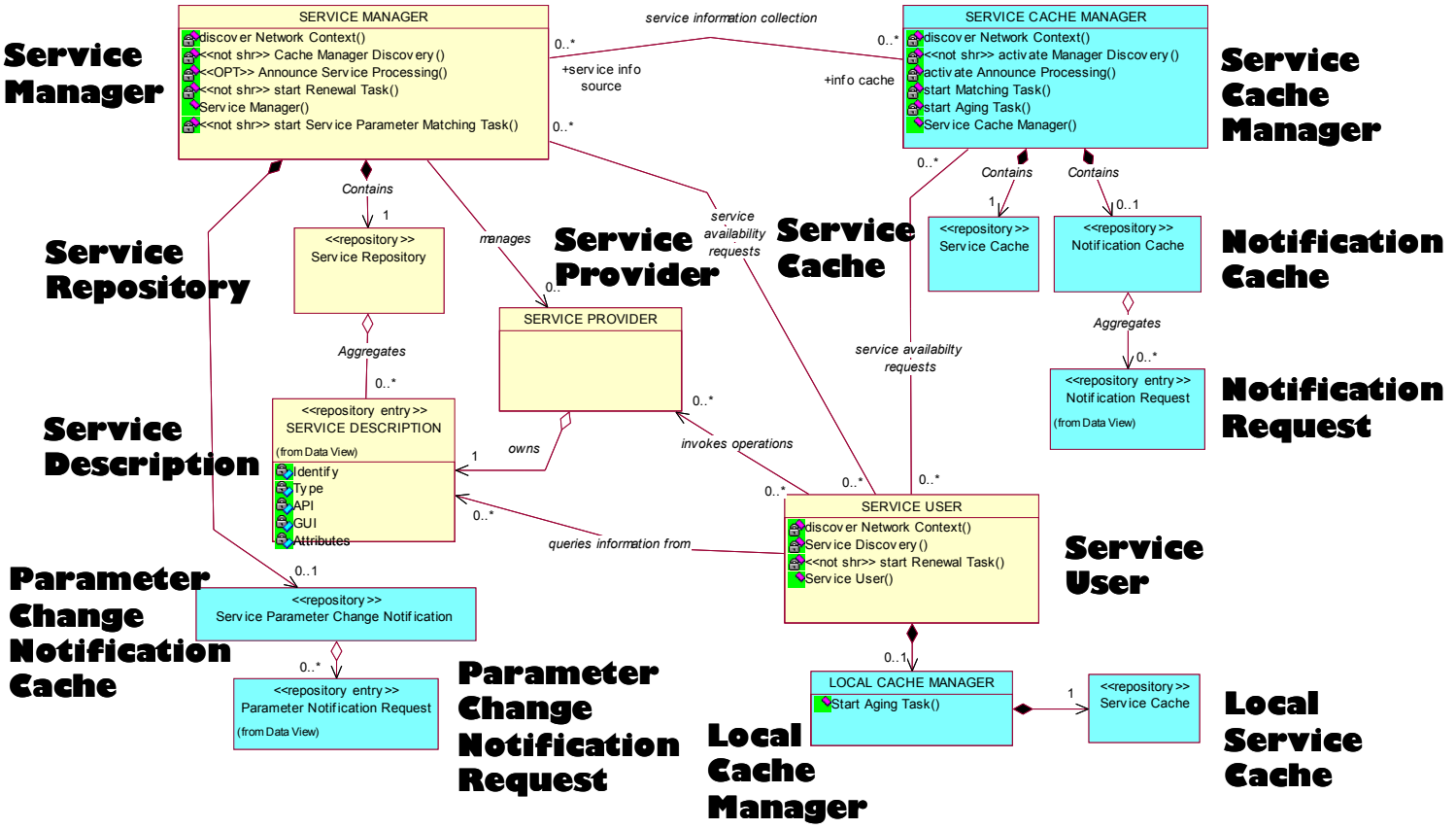
- **Model** Discovery Protocol **specifications using Architectural Description Languages** (ADLs) and associated tools
- **Analyze** Discovery Protocol **models** to assess consistency, correctness, and completeness under conditions of dynamic change.
- **Compare and contrast** our **models** with regard to function, structure, behavior, performance, complexity, and scalability under conditions of dynamic change.

# Technical Approach to Measurement

- Design technology-independent **benchmark service and scenarios**.
- Create **synthetic workload generation tools** for emulating the behavior of moderate-scale dynamic ad hoc networking environments.
- Develop implementation-independent **performance measurement methodologies and tools** for service discovery protocols (SDPs) and required supporting protocols.

**EXAMPLES FOLLOW ON NEXT THREE SLIDES**

# Example #1: Define Generic Structural Model (UML) for Service-Discovery Domain



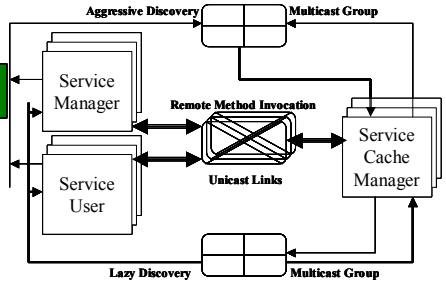

## Provides Foundation for Measurements and Comparisons

# Example #2: Construct, Exercise, and Analyze Executable Model from Specifications of Discovery Protocols

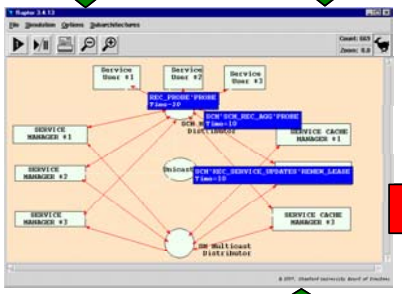
| Time | Command | Parameters |
|---|---|---|
| 5 | NodeFail | SM4 |
| 5 | LinkFail | SCM1 SM4 |
| 10 | GroupJoin | SM4 GROUP1 |
| 10 | FindService | SU8 5 1 2 S XYZ ALL |
| 50 | AddService | SM4 SCM3 T ATT API GUI 20 30 |

Scenario

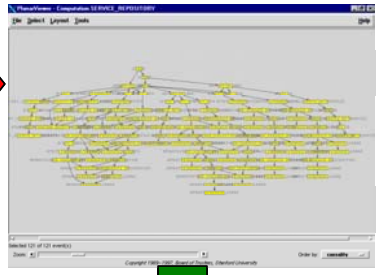
Topology




Specification Model



Execute with Rapide



```
-- ******************************************************
-- ** 3.3  DIRECTED DISCOVERY CLIENT INTERFACE     **
-- ******************************************************
-- This is used by all JINI entities in directed
-- discovery mode.  It is part of the SCM_Discovery
-- Module. Sends Unicast messages to  SCMs on list of
-- SCMS to be discovered until all SCMS are found.
-- Receives updates from SCM DB of discovered SCMs and
-- removes SCMs accordingly
-- NOTE: Failure and recovery behavior are not
-- yet defined and need reviw.
TYPE Directed_Discovery_Client
  (SourceID : IP_Address; InSCMsToDiscover : SCMList; StartOption : DD_Code;
   InRequestInterval : TimeUnit; InMaxNumTries : integer; InPV : ProtocolVersion)
IS INTERFACE
SERVICE DDC_SEND_DIR    : DIRECTED_2_STEP_PROTOCOL;
SERVICE DISC_MODES      : dual SCM_DISCOVERY_MODES;
SERVICE DD_SCM_Update   : DD_SCM_Update;
SERVICE SCM_Update      : SCM_Update;
SERVICE DB_Update       : dual DB_Update;
SERVICE NODE_FAILURES : NODE_FAILURES;  -- events for failure and recovery.
ACTION
 IN Send_Requests(),
    BeginDirectedDiscovery();
BEHAVIOR
   action animation_Iam (name: string);
   MySourceID        : VAR IP_Address;
   PV                : VAR ProtocolVersion;
```

Consistency Conditions

**For All (SM, SD, SCM):**
**(SM, SD) IsElementOf SCM registered-services (CC1)**
**implies SCM IsElementOf SM discovered-SCMs**

**For All (SM, SD, SCM):**
**SCM IsElementOf SM discovered-SCMs & (CC2)**
**(SD) IsElementOf SM managed-services**
**implies (SM, SD) IsElementOf SCM registered-services**

**For All (SM, SD, SCM):**
**SCM IsElementOf SM discovered-SCMs & (CC3)**
**(SM, SD) IsElementOf SCM registered-services &**
**NOT (SCM IsElementOf SM persistent-list)**
**implies intersection (SM GroupsToJoin, SCM GroupsMemberOf)**

**For All (SM, SD, SCM, SU, NR):**
**(SU, NR) IsElementOf SCM requested-notifications & (CC4)**
**(SM, SD) IsElementOf SCM registered-services &**
**Matches((SM, SD), (SU,NR))**
**implies (SM, SD) IsElementOf SU matched-services**

Analyze POSETs

Assess Correctness, Performance, & Complexity

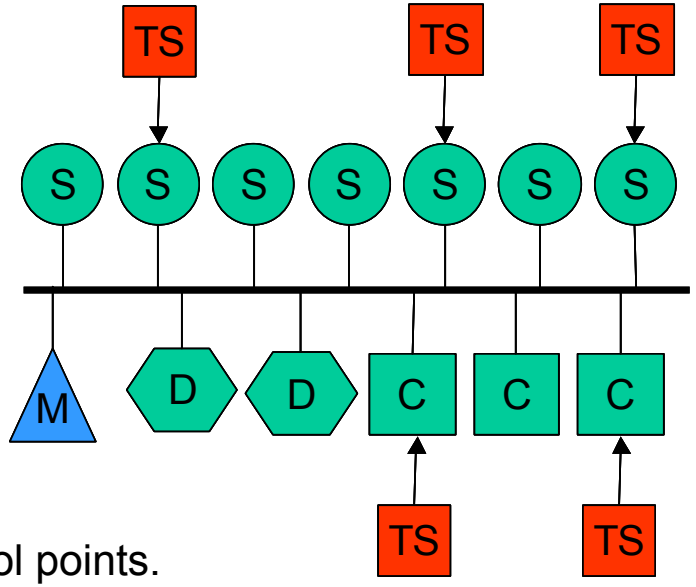
## Understand Logical and Performance Properties Arising from Collective Behavior

# Example #3: Construct Synthetic Workload Generation and Measurement Tools

**Objective:** Emulate large, dynamic environments of 100's of devices/services and 10's of control points / clients.

- Dynamic devices provided the benchmark service.
- Scripted control points execute measurement scenarios.

- **SDP Experimenters Toolkits**
  - Drive real SDP implementations
  - Emulate the behavior of a large number of dynamic devices
  - Emulate the behavior of control points and provide scripted behavior for testing
- **Jini & UPnP Initial development complete**
  - SunMS Jini, Intel UPnP on Linux platforms.
  - Achieved 100's of devices and 10's of control points.




## Measure Latency and Overhead in Moderate-Scale Deployment of Discovery Protocol Implementations

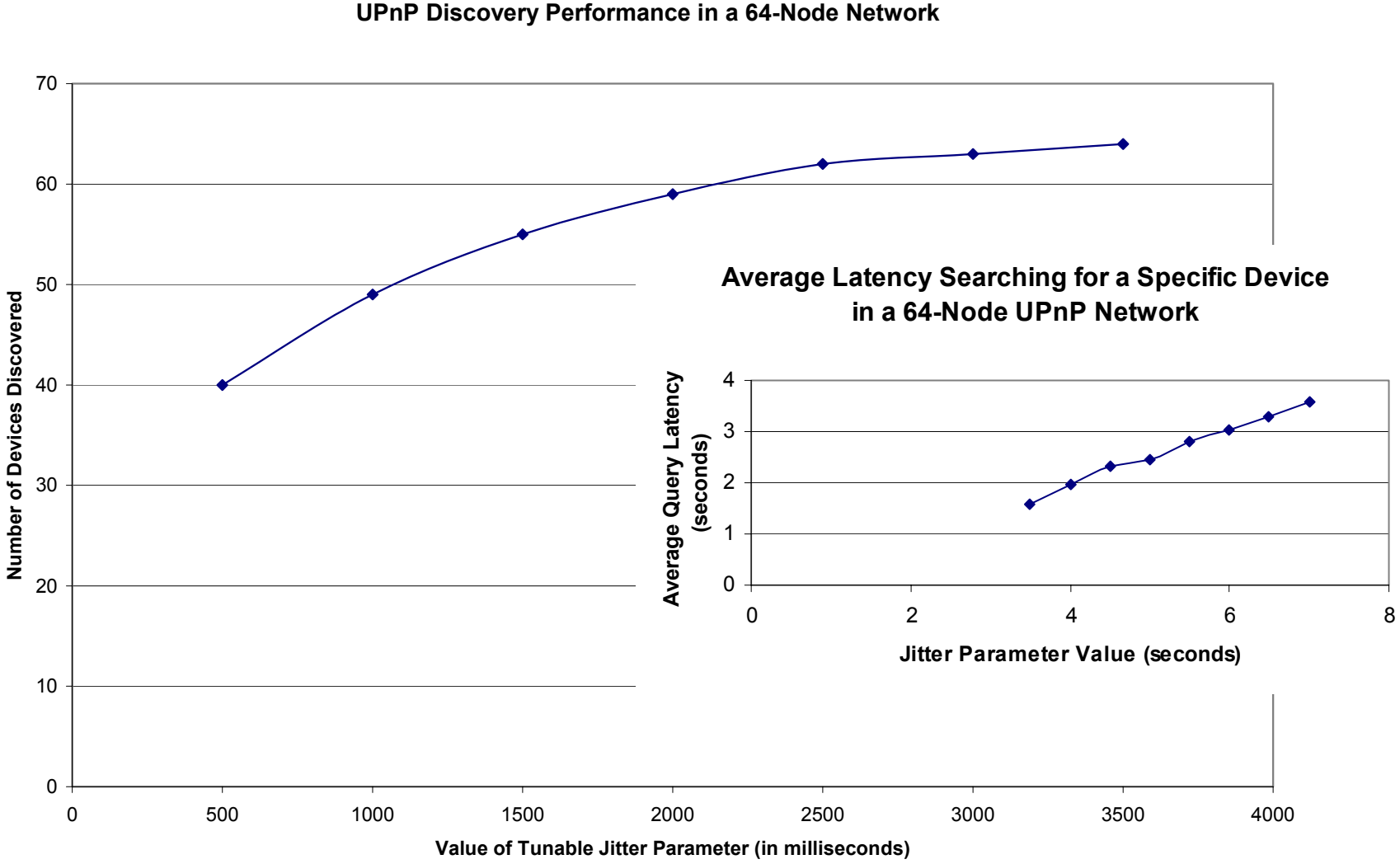
# Objective for Our FTN Project

*Research, design, evaluate, and implement self-adaptive algorithms to improve the performance of service discovery protocols for use in fault-tolerant networks.*

# Motivation

- Emerging designs for military fault-tolerant systems (e.g., OpenWings, OASIS, CoABS) rely on discovery-based component architectures to enable self-organizing and self-healing behavior
- The discovery protocols underlying such systems include mechanisms that permit network elements to continue to function as the topology varies
- However, many performance aspects of these protocols appear sensitive to parameter settings whose optimum values depend upon network topology
- While such parameters may be manually configured and tuned in relatively small, static environments, their management in larger scale, highly dynamic environments requires decentralized real-time measurement and control

# Sample Problem Based on Measurements of Universal Plug-and-Play



# Selected Control Parameters of Interest

- Universal Plug-and-Play
  - announcement interval
  - response jitter time
  - entry expiration time
  - requested and granted subscription expiration times
  - maximum response time
  - message repeat count and interval
- Jini™ Networking Technology
  - announcement and probe intervals
  - requested and granted lease expiration times
  - maximum lookup servers to discover
  - maximum matches returned
- Service Location Protocol
  - announcement and probe intervals
  - entry lifetime
  - minimum refresh interval
  - maximum response time
  - message repeat period and interval

# Possible Adaptive Behaviors of Interest

- Universal Plug-and-Play
  - dynamic reassignment of multicast group membership
  - response of device to changes in device descriptions
  - cache filtering, purging, and triggering in control points
  - query behavior in control points
- Jini™ Networking Technology
  - dynamic reassignment of logical group membership
  - dynamic reassignment of multicast group membership
  - injection of event filters into services
  - dynamic management of lookup services in response to load variation
- Service Location Protocol
  - dynamic mode switching between 2-party and 3-party operation
  - dynamic management of cache flushing by directory service agents
  - dynamic management of scopes

# Research Plan

- Model and analyze existing protocols (UPnP, Jini, and SLP)
  - develop simulation models for each protocol
  - determine appropriate techniques to model link and node failures
  - establish performance benchmarks based on default or recommended parameter values and on required or most likely implementation of behaviors
- Investigate distributed adaptation algorithms to control parameter values (and also consider selected adaptive behaviors)
  - devise several algorithms to adjust similar control parameters in each protocol
  - simulate performance of each algorithm against benchmark performance
  - select most promising algorithms for further development
- Implement and validate selected algorithms in publicly available reference software
  - modify available implementation of UPnP, Jini, or SLP
  - deploy in service-discovery test bed (now under development at NIST)
  - validate simulated results with live experiments
- Expected results: published papers, simulation models, reference software, experiment data, influence on design of future dynamic discovery protocols

# Conclusions

- Emerging designs for military fault-tolerant systems rely on discovery-based component architectures to enable self-organizing and self-healing behavior
- Emerging industry discovery protocols assume parameters configured and tuned by hand and network topology evolves slowly
  - Such assumptions are invalid for most military scenarios, which require fast response, exhibit high mobility, suffer cyber and physical attacks, as well as jamming of communication channels
- We propose a solution: read-and-react decentralized discovery mechanisms that observe the network state, estimate and adjust relevant parameters, measure effects, and adapt as necessary
  - How will such mechanisms perform under military and commercial conditions?
  - What will be the cost of such mechanisms in complexity and overhead?
  - Can existing commercial protocols accommodate such mechanisms?
- Success in this research would enable the military to leverage future generations of dynamic discovery protocols as a foundation for survivable software architectures